Universidade Federal do Mato Grosso
Instituto de Computação

PROVA DE TELEFAC

ANTHONY RICARDO RODRIGUES REZENDE
ALAN BRUNO MORAIS COSTA
VINICIUS PADILHA VIEIRA
ANDREY LUIGGI DA CRUZ

CUIABÁ, MATO GROSSO 2023

ANTHONY RICARDO RODRIGUES REZENDE
ALAN BRUNO MORAIS COSTA
VINICIUS PADILHA VIEIRA
ANDREY LUIGGI DA CRUZ

PROVA DE TELEFAC

Trabalho sobre "Resolução da Avaliação de TELE-FAC"conforme a disciplina de Teoria das Linguagens Formais e Autômatos, apresentado como requerido para a obtenção de nota na Universidade Federal de Mato Grosso - UFMT

Professor: Dr. Eduardo

CUIABÁ, MATO GROSSO 2023

Resumo

Apresentação da atividade avaliativa sobre Autômatos Finitos Determinísticos, no formato de relatório.

1 Introdução

A princípio este documento apresenta o relatório do processo de criação e testes do algoritmo para simular o reconhecimento de palavras com um AFD (Autômato Finito Determinístico), feito em C++. Ainda que no mesmo documento visa-se apresentar os desafios e problemas que encontramos no decorrer da construção do algoritmo.

2 Relatório de Implementação (Criação)

2.1 Estruturas de Dados

O código em C++ implementa um Autômato Finito Determinístico (AFD) aproveitando estruturas da biblioteca STL (Standard Template Library), cada uma selecionada por suas funcionalidades específicas.

- **Set:** é usado para armazenar estados finais e o alfabeto do AFD, aproveitando sua capacidade de manter elementos únicos e ordenados, facilitando a verificação de estados finais e símbolos do alfabeto.
- **Map:** representa a tabela de transição do AFD, mapeando pares estado-símbolo a estados subsequentes, garantindo relações chave-valor únicas para a lógica de transição.
- **Vector:** é mencionado mas não utilizado explicitamente. Serve para armazenar elementos em sequência, permitindo acesso rápido e manipulação eficiente.
- **String:** é empregado para armazenar a cadeia de entrada testada pelo AFD, permitindo manipulação e operações em textos.
- **Pair:** é usado nas chaves do std::map para a tabela de transição, combinando estado e símbolo em um objeto único, essencial para mapear as transições do AFD.

  Essas estruturas são escolhidas por suas propriedades que atendem aos requisitos do AFD, como unicidade de elementos (set), mapeamento chave-valor (map), e manipulação de textos (string).

2.2 Algoritmos Implementados

Este código em C++ representa um Autômato Finito Determinístico (AFD), dividido em definição da classe AFD e função main.

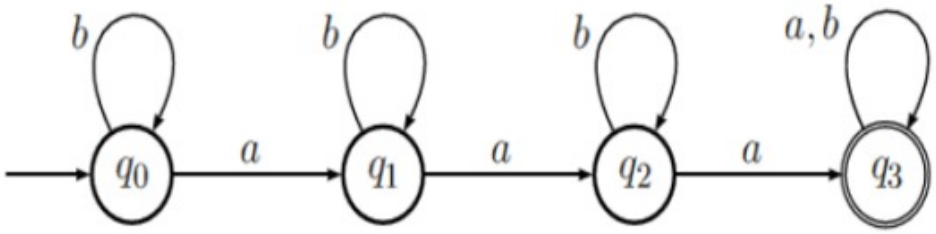
- **Classe AFD:**
  - **Atributos Privados:**
    - **numEstados:** Total de estados do autômato.
    - **estadoInicial:** Estado inicial.
    - **estadosFinais:** Conjunto dos estados de aceitação.
    - **alfabeto:** Conjunto dos símbolos do alfabeto. tabelaTransicoes: Mapa do par estado-símbolo para estado seguinte.
  - **Métodos:**
    - **AFD():** Construtor da classe.
    - **lerAFD():** Lê componentes do AFD da entrada padrão. aceita(string entrada): Determina se a cadeia é aceita pelo autômato.
- **Função main:** Instancia a classe AFD, lê sua configuração e uma cadeia de entrada, e usa aceita para avaliar a cadeia, imprimindo "Aceito"ou "Rejeitado". O programa inicia lendo a configuração do AFD, incluindo estados, estado inicial, estados finais, alfabeto e transições, e depois lê uma cadeia de entrada do usuário. Utiliza aceita para verificar se a cadeia é aceita, baseado nas transições e se termina em estado final, determinando o resultado como "Aceito"ou "Rejeitado".

3 AFD - 01

Abaixo está o primeiro Autômato Finito Determinístico para análise, e suas respectivas características e representação gráfica. Algoritmo foi testado usando **MinGW** na **IDE VsCode**.

L(M) = { x ∈ $\{b^*ab^*ab^*a(a+b)^*\}$: garantindo no mínimo 3 letras $a\}$

Figura 1: AFD 01



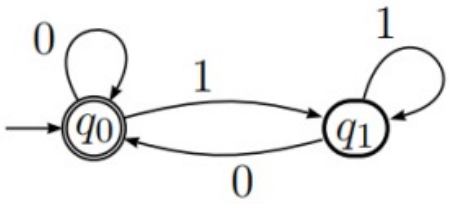
Fonte: print do computador

4 AFD - 02

Abaixo está o segundo Autômato Finito Determinístico para análise, e suas respectivas características e representação gráfica. Algoritmo foi testado usando **MinGW** na **IDE VsCode**.

Figura 2: AFD 02

$L(M) = \{x \in \{0,1\}^* : x$ representa um número par$\}$.



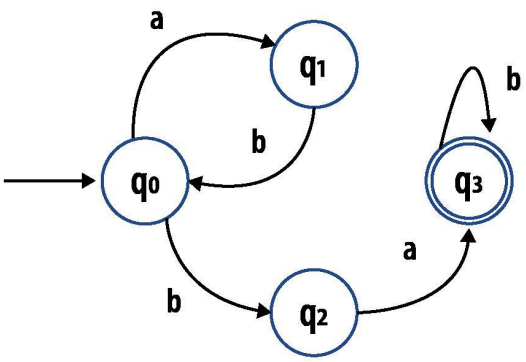
Fonte: print do computador

5 AFD - 03

Abaixo está o terceiro Autômato Finito Determinístico para análise, e suas respectivas características e representação gráfica. Algoritmo foi testado usando **MinGW** na **IDE VsCode**.

L(M) = { x ∈ $\{(a\ |b)^*a(a\ |b)^*\}$: garantindo no mínimo 3 letras $a\}$

Figura 3: AFD 03



Fonte: print do computador

6 Dificuldades Encontradas

Ao desenvolver um algoritmo para simular um autômato finito determinístico (AFD) e aplicá-lo para reconhecer cadeias de caracteres,a maior dificuldade foi a transição que dividiu opiniões, como por exemplo o caminho duplo na figura 1, q3 a,b q3, criamos dois caminhos, q3 a q3, q3 b q3, o ideal seria criar um caminho único, como citado anteriormente.

A implementação da função que reconhece se uma cadeia de entrada é aceita pelo AFD. Essa função precisa acompanhar o estado atual do AFD conforme lê cada símbolo da cadeia de entrada e determinar corretamente se a cadeia leva a um estado de aceitação. A complexidade nasce ao garantir que todas as transições estejam corretamente definidas e que o algoritmo trate corretamente cadeias que não são reconhecidas.

- **Seleção de Autômatos para Teste:** Determinar quais autômatos usar como exemplos ou para testar o simulador pode ser complicado. Isso requer um entendimento de como diferentes autômatos funcionam e quais tipos de linguagens eles são capazes de reconhecer. A escolha de exemplos representativos é crucial para validar a abrangência e a correção do simulador.

7 Aprendizados Adquiridos

- **Conhecimentos Teóricos:**
  - **Linguagens Formais:** Aprender sobre a classificação de linguagens (regulares, livres de contexto, etc.) e a relação entre autômatos e linguagens formais, incluindo como linguagens podem ser definidas por expressões regulares e gramáticas formais.
- **Habilidades Práticas**
  - **Desenvolvimento de Algoritmos:** Praticar conceitos teóricos em códigos funcionais, desenvolvendo habilidades de programação e algorítmicas. Aprende a lidar com estruturas de dados, como grafos (que podem representar autômatos), e a manipular cadeias e realizar buscas eficientes.
  - **Resolução de Problemas:** Ao implementar o simulador e resolver problemas relacionados a autômatos, podemos melhorar nossas habilidades de resolução de problemas, aprendendo a decompor problemas complexos em subproblemas mais gerenciáveis.

8 Conclusão

Este trabalho evidenciou a importância e a complexidade da implementação de Autômatos Finitos Determinísticos (AFD) utilizando a linguagem C++. Ao superar desafios técnicos e teóricos, a equipe adquiriu conhecimentos profundos sobre estruturas de dados, algoritmos, e a teoria subjacente às linguagens formais e autômatos. Os testes realizados com diferentes AFDs demonstraram a eficácia do simulador desenvolvido, validando a abordagem adotada e reforçando a capacidade de aplicar conceitos teóricos em situações práticas. Este

projeto não apenas cumpriu seus objetivos educacionais, mas também forneceu uma base sólida para futuras investigações e desenvolvimentos na área de ciência da computação, especialmente no estudo de linguagens formais e teoria da computação.